I217: Functional Programming

9. A Programming Language Processor – Virtual Machine

Kazuhiro Ogata

Roadmap

- Virtual Machine

Virtual Machine

The virtual machine has a set of instructions.

One of the instructions is quit.

Given a list of instructions, it executes the instruction list with a program counter, a stack of natural numbers and an environment and returns the environment at the time when the virtual machine encounters the instruction quit.

It may return errEnv if something wrong, such as division by zero, occurs.

Virtual Machine

The virtual machine repeats the following until it encounters the instruction quit:

- ✓ It fetches the instruction pointed by the program counter.
- ✓ It modifies the stack, the environment and/or the program counter based on the instruction.

When it encounters quit, it returns the environment.

If something wrong, such as division by zero, happens, it returns errEnv.

Virtual Machine

Given the list of instructions

push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply |
store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) |
equal | jumpOnCond(2) | bjump(12) | quit | iln .

the virtual machine returns the environment

((x , 64) | ((y , 56636) | empEnv)):Env

The list of instructions is a program that calculates 2^{16} and stores the result in y.

Virtual Machine

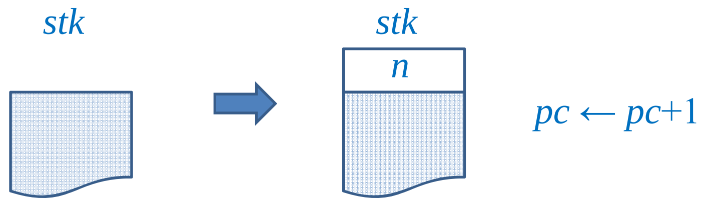
The instructions of the virtual machine:

push(*n*)	load(*x*)	store(*x*)	add
minus	multiply	divide	mod
lessThan	greaterThan	equal	notEqual
and	or	jump(*n*)	bjump(*n*)
jumpOnCond(*n*)	quit		

Virtual Machine

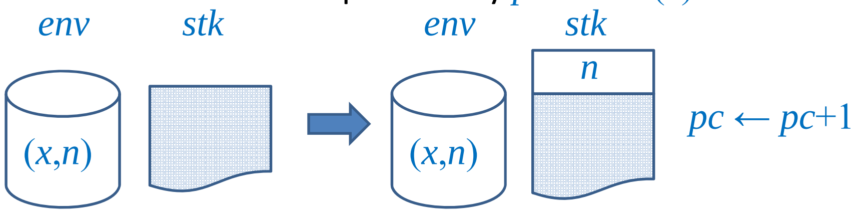
Let *pc*, *stk* & *env* be the program counter, the stack & the environment used in the virtual machine (vm).

✓ When the instruction pointed by *pc* is push(*n*)

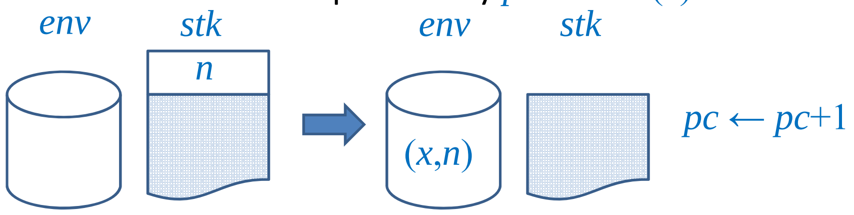


Virtual Machine

✓ When the instruction pointed by *pc* is load(*x*)



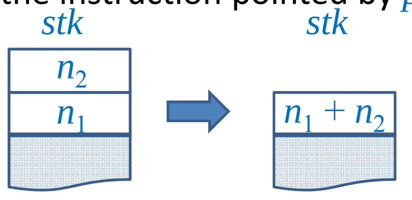
✓ When the instruction pointed by *pc* is store(*x*)



If *stk* is empty, the vm returns errEnv.

Virtual Machine

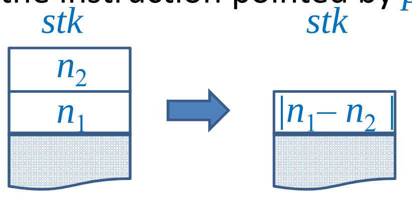
- ✓ When the instruction pointed by *pc* is add



$pc \leftarrow pc+1$

If *stk* has one or zero element, the vm returns errEnv.

- ✓ When the instruction pointed by *pc* is minus

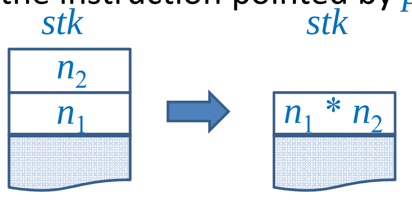


$pc \leftarrow pc+1$

If *stk* has one or zero element, the vm returns errEnv.

Virtual Machine

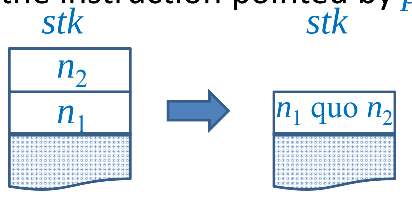
- ✓ When the instruction pointed by *pc* is multiply



$pc \leftarrow pc+1$

If *stk* has one or zero element, the vm returns errEnv.

- ✓ When the instruction pointed by *pc* is divide

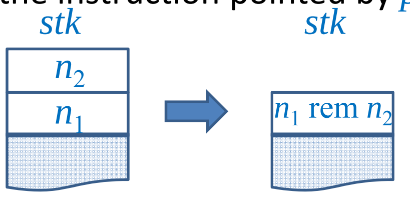


$pc \leftarrow pc+1$

If *stk* has one or zero element, the vm returns errEnv.

Virtual Machine

✓ When the instruction pointed by *pc* is mod

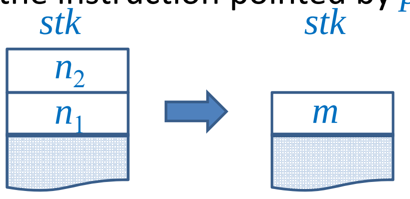


$pc \leftarrow pc+1$

If *stk* has one or zero element, the vm returns errEnv.

Virtual Machine

✓ When the instruction pointed by *pc* is lessThan



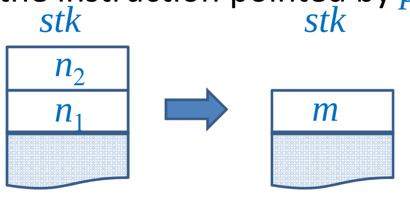
$pc \leftarrow pc+1$

$$m = \begin{cases} 1 & \textbf{if } n_1 < n_2 \\ 0 & \textbf{otherwise} \end{cases}$$

If *stk* has one or zero element, the vm returns errEnv.

Virtual Machine

- ✓ When the instruction pointed by *pc* is greaterThan



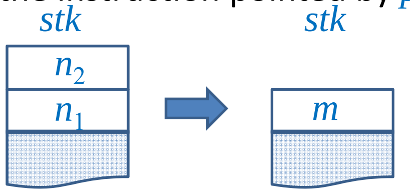
$pc \leftarrow pc{+}1$

$$m = \begin{cases} 1 & \textbf{if } n_1 > n_2 \\ 0 & \textbf{otherwise} \end{cases}$$

If *stk* has one or zero element, the vm returns errEnv.

Virtual Machine

- ✓ When the instruction pointed by *pc* is equal



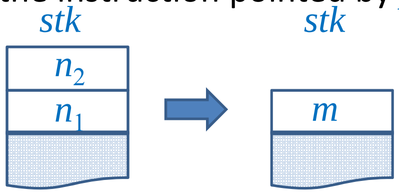
$pc \leftarrow pc{+}1$

$$m = \begin{cases} 1 & \textbf{if } n_1 = n_2 \\ 0 & \textbf{otherwise} \end{cases}$$

If *stk* has one or zero element, the vm returns errEnv.

Virtual Machine

✓ When the instruction pointed by pc is notEqual



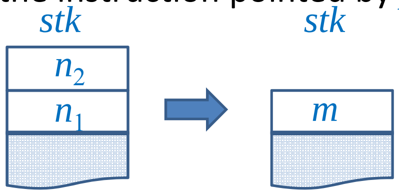
$pc \leftarrow pc+1$

$$m = \begin{cases} 1 & \textbf{if } n_1 \neq n_2 \\ 0 & \textbf{otherwise} \end{cases}$$

If stk has one or zero element, the vm returns errEnv.

Virtual Machine

✓ When the instruction pointed by pc is and



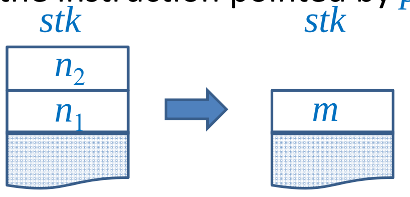
$pc \leftarrow pc+1$

$$m = \begin{cases} 1 & \textbf{if } n_1 \neq 0 \text{ and } n_2 \neq 0 \\ 0 & \textbf{otherwise} \end{cases}$$

If stk has one or zero element, the vm returns errEnv.

Virtual Machine

✓ When the instruction pointed by *pc* is or



$pc \leftarrow pc+1$

$$m = \begin{cases} 1 & \textbf{if } n_1 \neq 0 \text{ and/or } n_2 \neq 0 \\ 0 & \textbf{otherwise} \end{cases}$$

If *stk* has one or zero element, the vm returns errEnv.

Virtual Machine

✓ When the instruction pointed by *pc* is jump(n)

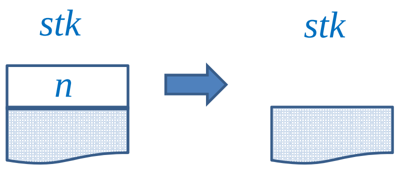
$\Rightarrow$ $pc \leftarrow pc+n$

✓ When the instruction pointed by *pc* is bjump(n)

$\Rightarrow$ $pc \leftarrow pc - n$

Virtual Machine

✓ When the instruction pointed by *pc* is junpOnCond(*n*)

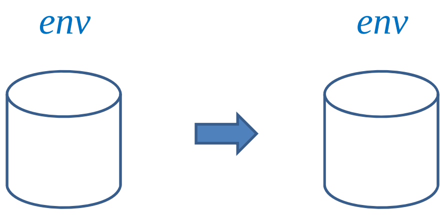


$$pc \leftarrow \begin{cases} pc+n & \textbf{if } n \neq 0 \\ pc+1 & \textbf{otherwise} \end{cases}$$

If *stk* is empty, the vm returns errEnv.

Virtual Machine

✓ When the instruction pointed by *pc* is quit



is returned as the result.

Virtual Machine

```
op run : IList -> Env&Err .
op exec : IList Nat Stack&Err Env&Err -> Env&Err .
op exec2 : Instruct&Err IList Nat Stack&Err Env&Err -> Env&Err .

var IL : IList .  var PC : Nat .  var Stk : Stack .  var Env : Env .
vars N N1 N2 : Nat .  var V : Var .  var E&E : Env&Err .
var S&E : Stack&Err .  var I&E : Instruct&Err .
```

a list of instructions   *stk* that is initially empty

```
eq run(IL) = exec(IL,0,empstk,empEnv) .
```

pc that is initially 0   *env* that is initially empty

Virtual Machine

```
eq exec(IL,PC,errStack,E&E) = errEnv .
eq exec(IL,PC,S&E,errEnv) = errEnv .
eq exec(IL,PC,Stk,Env) = exec2(nth(IL,PC),IL,PC,Stk,Env) .
```

If *stk* is errStack and/or *env* is errEnv, then exec returns errEnv.

Otherwise, exec fetches the instruction nth(IL,PC) pointed by *pc* and modify *pc*, *stk* and/or *env* with exec2.

If PC is out of the range of IL, then nth(IL,PC) becomes errInstruct.

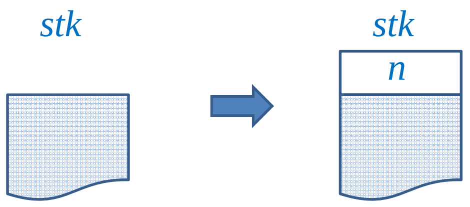
Virtual Machine

op exec2 : Instruct&Err IList Nat Stack&Err Env&Err -> Env&Err .

If the instruction is errInstruct, *stk* is errStack and/or *env* is errEnv, then exec2 returns errEnv.

✓ When the instruction pointed by *pc* is push(*n*)

eq exec2(push(N),IL,PC,Stk,Env) = exec(IL,PC + 1,N | Stk,Env) .

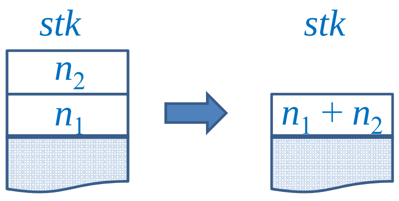


$pc \leftarrow pc+1$

Virtual Machine

✓ When the instruction pointed by *pc* is add

eq exec2(add,IL,PC,empstk,Env) = errEnv .
eq exec2(add,IL,PC,N1 | empstk,Env) = errEnv .
eq exec2(add,IL,PC,N2 | N1 | Stk,Env)
    = exec(IL,PC + 1,N1 + N2 | Stk,Env) .



$pc \leftarrow pc+1$

If *stk* has one or zero element, the vm returns errEnv.

For the remaining instructions, equations can be described likewise for exec2.

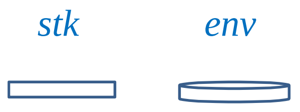
Virtual Machine

Let *il* be

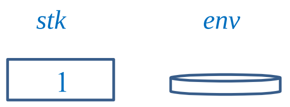
push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

Let us consider

exec(*il*,0,empstk,empEnv)

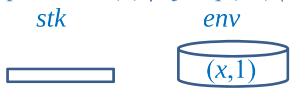


1. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

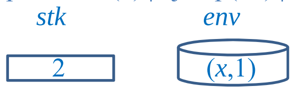


Virtual Machine

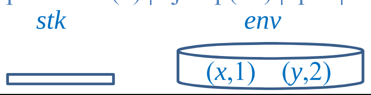
2. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

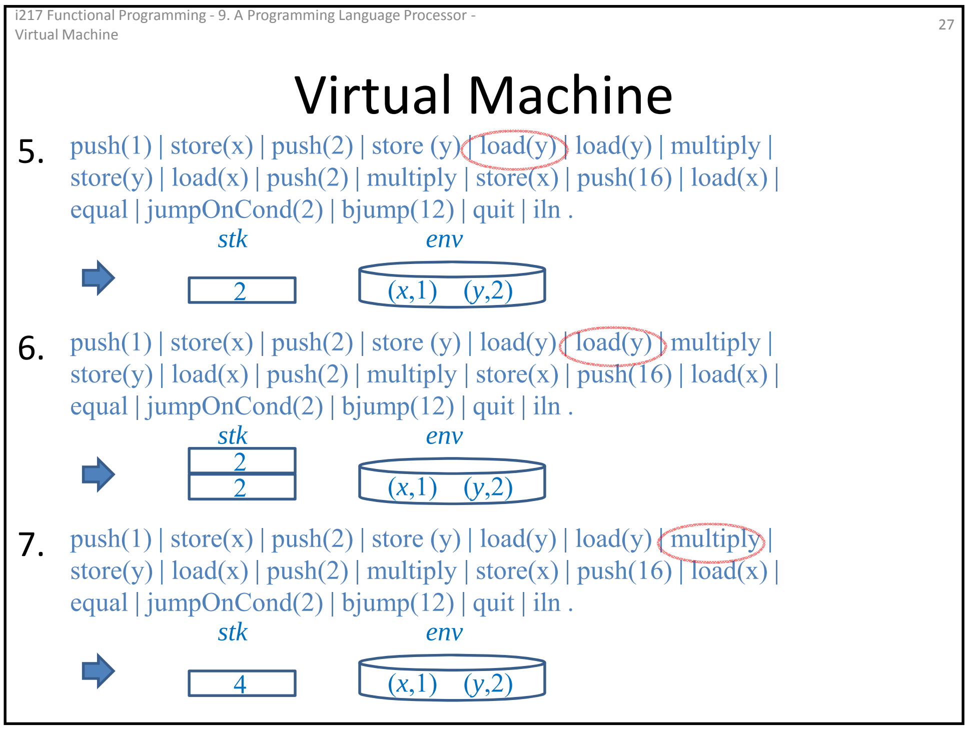


3. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



4. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .





Virtual Machine

5. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

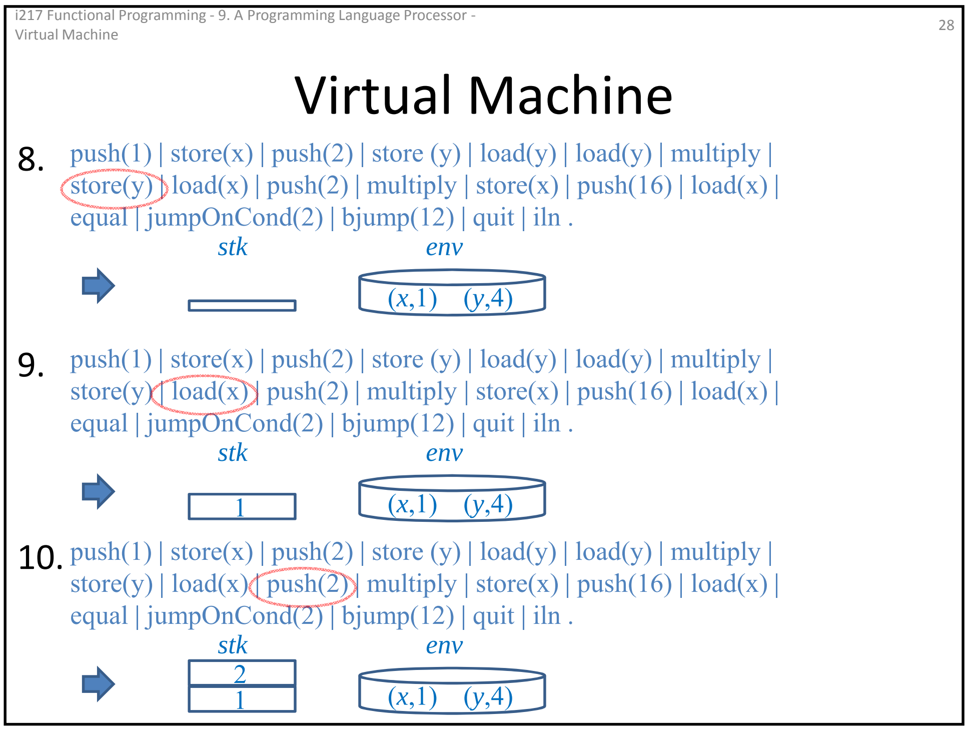
stk 2 | *env* (*x*,1) (*y*,2)

6. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

stk 2 2 | *env* (*x*,1) (*y*,2)

7. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

stk 4 | *env* (*x*,1) (*y*,2)



Virtual Machine

8. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

stk | *env* (*x*,1) (*y*,4)

9. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

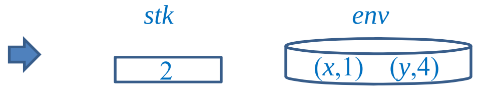
stk 1 | *env* (*x*,1) (*y*,4)

10. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

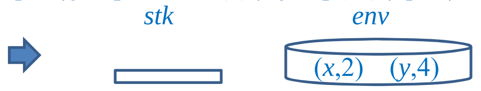
stk 2 1 | *env* (*x*,1) (*y*,4)

Virtual Machine

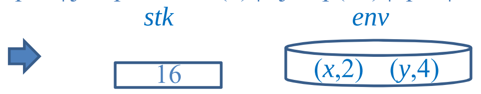
11. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



12. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

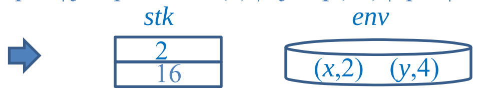


13. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

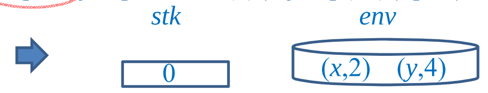


Virtual Machine

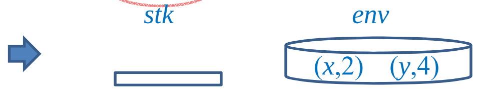
14. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

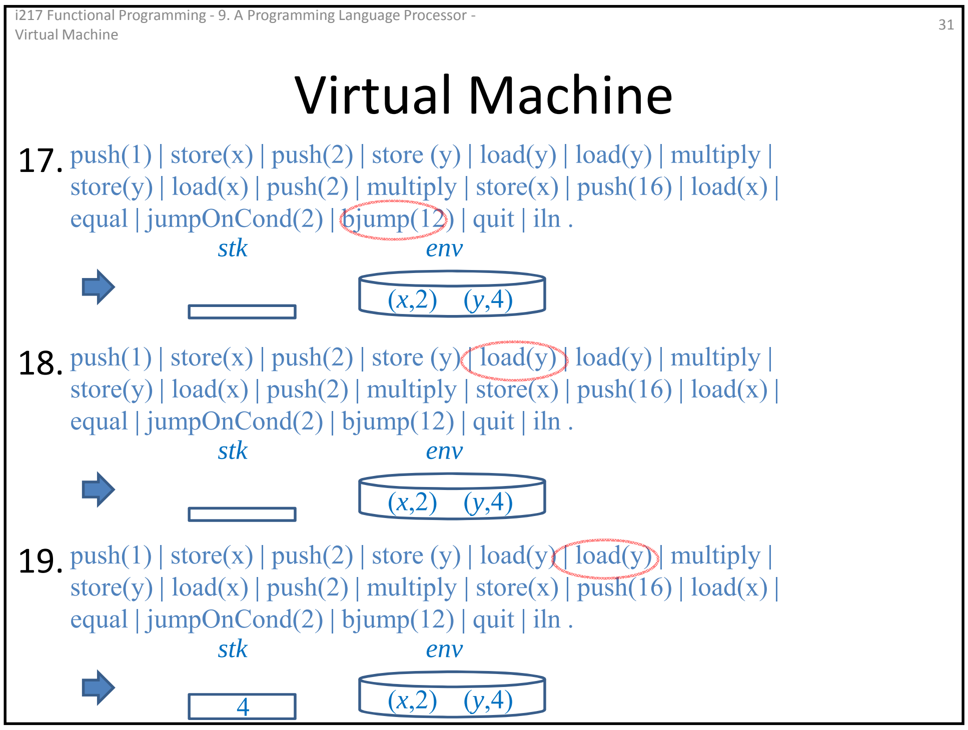


15. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



16. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



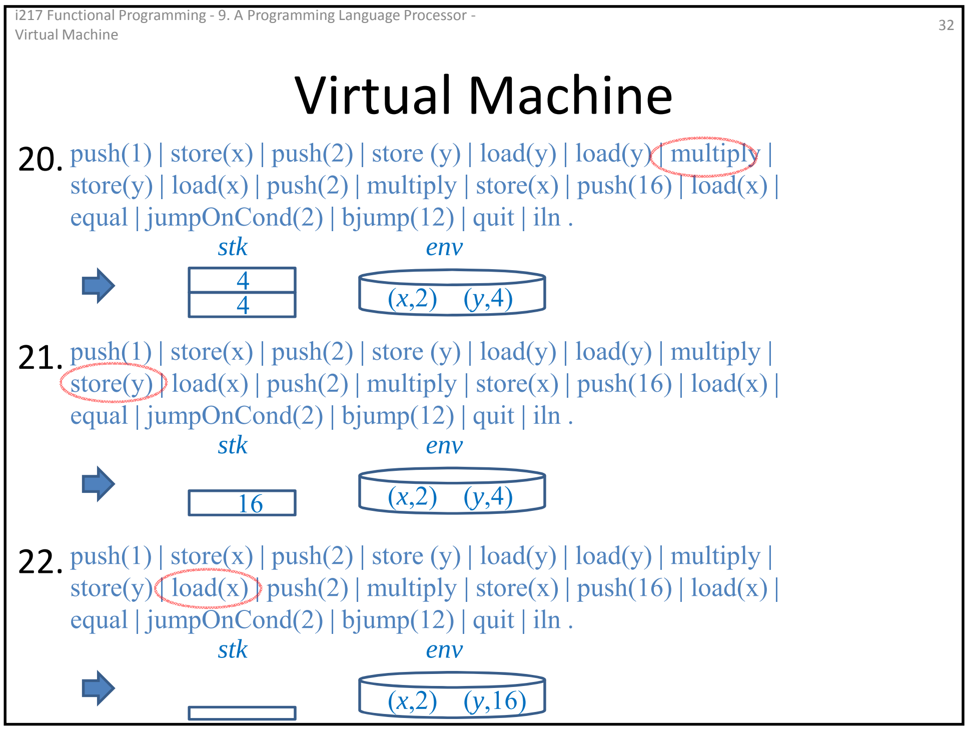


Virtual Machine

17. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply |
store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) |
equal | jumpOnCond(2) | bjump(12) | quit | iln .

18. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply |
store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) |
equal | jumpOnCond(2) | bjump(12) | quit | iln .

19. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply |
store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) |
equal | jumpOnCond(2) | bjump(12) | quit | iln .



Virtual Machine

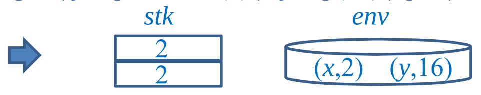
20. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply |
store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) |
equal | jumpOnCond(2) | bjump(12) | quit | iln .

21. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply |
store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) |
equal | jumpOnCond(2) | bjump(12) | quit | iln .

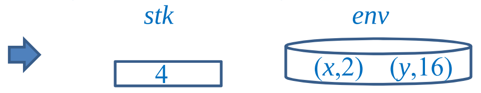
22. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply |
store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) |
equal | jumpOnCond(2) | bjump(12) | quit | iln .

Virtual Machine

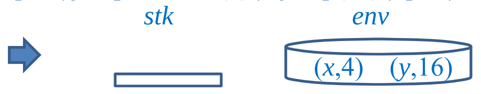
23. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



24. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

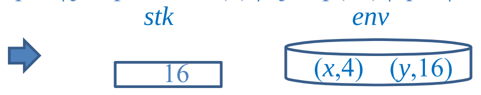


25. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

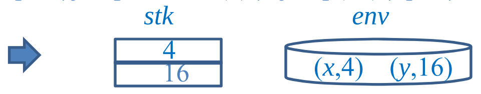


Virtual Machine

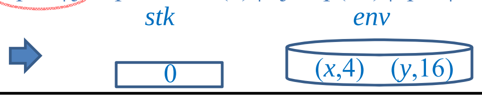
26. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



27. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

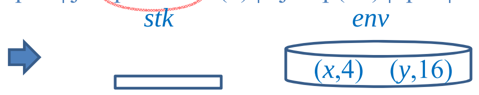


28. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

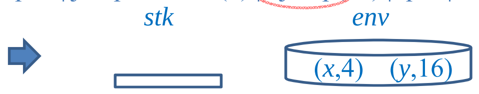


Virtual Machine

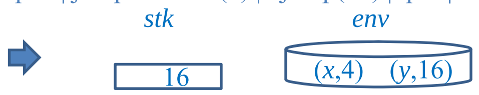
29. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



30. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

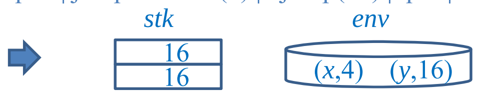


31. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

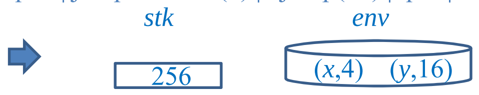


Virtual Machine

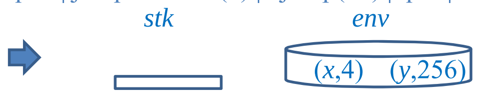
32. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



33. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

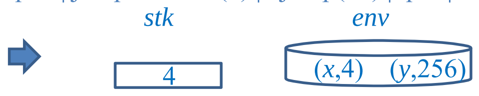


34. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

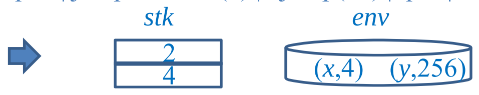


Virtual Machine

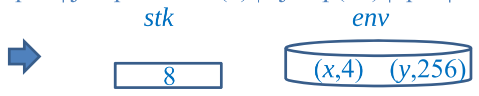
35. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



36. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

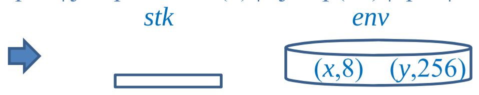


37. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

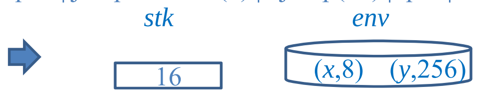


Virtual Machine

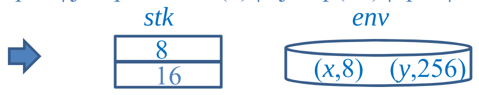
38. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



39. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

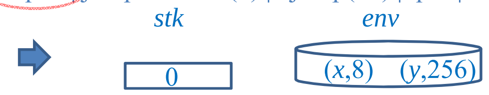


40. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

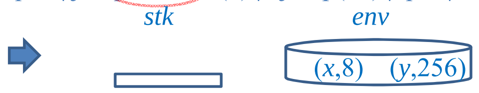


Virtual Machine

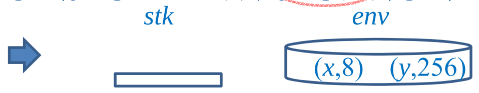
41. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



42. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

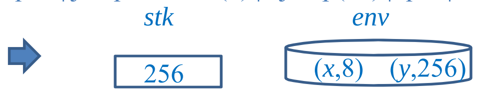


43. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

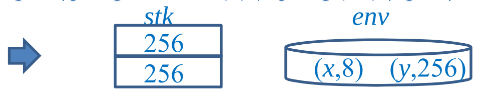


Virtual Machine

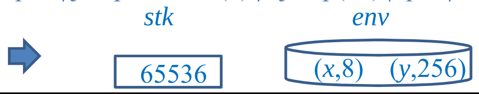
44. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



45. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

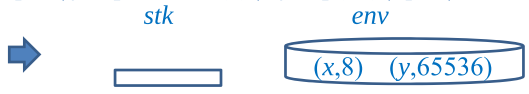


46. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

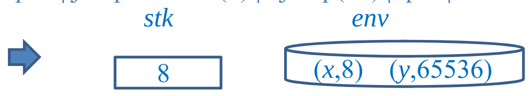


Virtual Machine

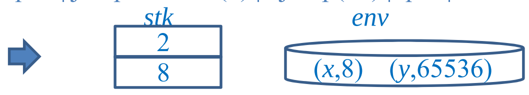
47. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



48. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

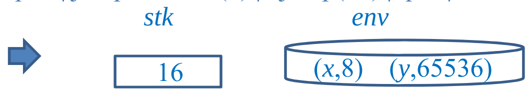


49. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

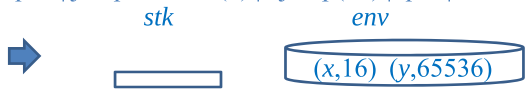


Virtual Machine

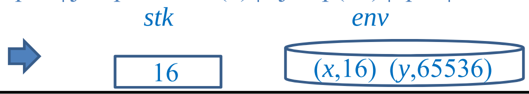
50. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



51. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

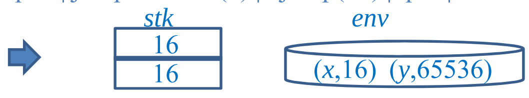


52. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

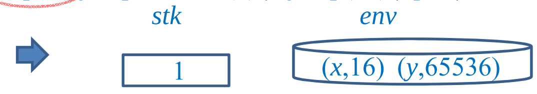


Virtual Machine

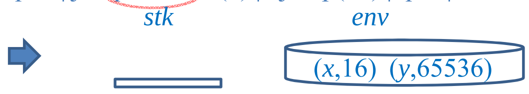
53. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



54. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .

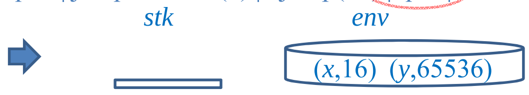


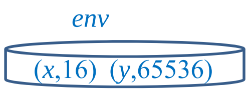
55. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .



Virtual Machine

56. push(1) | store(x) | push(2) | store (y) | load(y) | load(y) | multiply | store(y) | load(x) | push(2) | multiply | store(x) | push(16) | load(x) | equal | jumpOnCond(2) | bjump(12) | quit | iln .





is returned as the result.

Exercises

1. Complete the virtual machine and do some tests for the virtual machine.